MULTIMEDIA UNIVERSITY

TABLE NO | | |

STUDENT ID NO | | | | | | | | | | |

# MULTIMEDIA UNIVERSITY

# FINAL EXAMINATION

**TRIMESTER 2, 2022/2023 (TERM ID 2220)**

## TSE3151 – SOFTWARE DESIGN

(All sections / Groups)

7 JULY 2023
9:00 a.m – 11:00 a.m
(2 Hours)

| Question | Mark |
|---|---|
| **A** | |
| **B** | |
| **C** | |
| **D** | |
| Total | |

Examiner 1 Signature: _____

Examiner 2 Signature: _____

Examiner 3 Signature: _____

---

**INSTRUCTIONS TO STUDENTS**

1. This question paper consists of **11 printed pages** (including cover page).
2. Answer **ALL** questions in **SECTION A, SECTION B, SECTION C and SECTION D.** The distribution of the marks for each question is given.
3. Write all your answers in the **CLEARLY** in the specific answer box provided for each question. Submit this question paper at the end of the examination.

## STRUCTURED QUESTIONS (50 MARKS)

## SECTION A [15 MARKS]

Consider the following **SingletonPatternEx.java** program.

```
// SingletonPatternEx.java
package singleton.pattern.demo;

class MakeALeader
{
    private static MakeALeader icaptain;
    //We make the constructor private to prevent the use of "new"
    private MakeALeader() { }

  private static class SingletonHelper{
  //Nested class is referenced after getCaptain() is called
     private static final MakeALeader icaptain = new MakeALeader();
  } //end class SingletonHelper

  public static MakeALeader getCaptain()
  {
     return SingletonHelper.icaptain;
  }
} //end class MakeALeader

class SingletonPatternEx
{
    public static void main(String[] args)
  {
            System.out.println("***Singleton Pattern Potter***\n");
            System.out.println("Trying to make a leader for our team");
            //Put your code here
            }
}
```

Based on the above context, answer the following questions Q-A1 to Q-A3:

A1. **Rewrite the main method** of **SingletonPatternEx.java** to produce the following output by **creating two instances of singleton** and **comparing** whether they are the same instance.

```
F:\YourName>javac singleton\pattern\demo\SingletonPatternEx.java
F:\YourName>java singleton.pattern.demo.SingletonPatternEx
        ***Singleton Pattern Potter***
        Trying to make a leader for our team
        Trying to make another leader for our team
        leader1 and leader2 are same instance
```

**(5 marks)**

**Continued...**

**Answer**

A2. **Draw** a **Class Diagram** to show TWO relationships within these two classes, plus the attributes and methods. In your class diagram, please **write clearly what kind of relationships they represent**, *whether they are inheritance, dependency, implementation, aggregation, or composition (strong aggregation).*

**(5 marks)**

**Answer**

**Continued...**

A3. There are TWO situations concerning what we need to do to incorporate thread safety in the above Singleton implementation.
A3a. For the first situation, explain how we use the "synchronized" keyword, and its disadvantage.

**(1 mark)**

| **Answer** |
| --- |
| |

A3b. For the second situation, explain the THREE steps of eager initialization for class **MakeALeader** with the instance of ***icaptain*** to achieve thread safety. What is its advantage?

**(3 + 1 mark)**

| **Answer** |
| --- |
| |

**Continued...**

## SECTION B [15 MARKS]

Among the name in the **Factory Method** Design Pattern (DP) include *Creator*, *ConcreteCreator*, *factoryMethod()*, *Product*, *ConcreteProductA*, *ConcreteProductB*, *ConcreteProductC*, *myMethod()*, and *Client*.

Based on the above context, answer the following questions Q-B1 to Q-B2.

B1. Write a table to show the mapping of **Factory Method** DP to the example given in the following code.

```
package factory.pattern.demo;
interface IAnimal
{
        void speak();
}
class Duck implements IAnimal
{
        @Override
        public void speak()
        {       System.out.println("Duck says Pack-pack");
        }
}
abstract class IAnimalFactory
{
        public abstract IAnimal GetAnimalType(String type) throws Exception;
/* if we cannot instantiate later we'll throw exception */
}
class ConcreteFactory extends IAnimalFactory
{
        @Override
        public  IAnimal GetAnimalType(String type) throws Exception
        {
                switch (type)
                {
                case "Duck":
                        //duck= new Duck();
                        //return duck;
                        return new Duck();
                case "Tiger":
                        return new Tiger();
                case "Lion":
                        return new Lion();
                default:
                        throw new Exception( "Animal type : "+type+" cannot be
instantiated");
                }
        }
}
```

```
class Tiger implements IAnimal
{
        @Override
        public void speak()
        {       System.out.println("Tiger says: Halum..Halum");
        }
}
class Lion implements IAnimal
{
        @Override
        public void speak()
        {       System.out.println("Lion says: Roar..Roar!!");
        }
}
class FactoryPatternEx
{
        public static void main(String[] args) throws Exception
        {
                System.out.println("***Factory Pattern Demo***\n");
                IAnimalFactory animalFactory = new ConcreteFactory();
                IAnimal DuckType=animalFactory.GetAnimalType("Duck");
                DuckType.speak();
                IAnimal TigerType = animalFactory.GetAnimalType("Tiger");
                TigerType.speak();
                IAnimal LionType = animalFactory.GetAnimalType("Lion");
                LionType.speak();
        }
}
```

**(5 marks)**

**Answer**

**Continued...**

B2. Draw a class diagram for the typical Factory Method DP with the name in DP of *Creator*, *ConcreteCreator*, *factoryMethod( )*, *Product*, and *ConcreteProduct*.

**(6 marks)**

**Answer**

**Continued...**

B3. Draw a class diagram for the typical **Decorator** DP with the name in DP of *Component*, *ConcreteComponent*, *Decorator*, componentAggregationVariable, *operation( )*, *ConcreteDecoratorA*, and *ConcreteDecoratorB*. Please represent the relationships of implementation, inheritance, and aggregation clearly in your class diagram.

**(4 marks)**

| **Answer** |
| --- |
| |

**Continued...**

## SECTION C (10 MARKS)

C1. You are asked to plan a journey by air from Kuala Lumpur in Malaysia to Beijing in China.
Airline A is cheap, but involves flying to Bangkok, waiting there for five (5) hours and then flying on to Beijing. Airline B is the most expensive but offers a direct flight. Airline C has a package that is cheaper than that of airline A but involves flying from Kuala Lumpur to Taipei and then to Beijing via yet another airport in Taiwan.

**List FOUR factors** besides price which will affect your choice of airline? These choices would subsequently be focused on the airline software design.

**(2 marks)**

| **Answer** |
| --- |
| |

C2. Describe EIGHT criteria that qualifies software design as a wicked problem. *Note that "wicked problems" help to form the foundation principles for a software designer to be critique for knowing what to look for when a possible design is given.*

**(8 marks)**

| **Answer** |
| --- |
| |

**Continued...**

## SECTION D [10 MARKS]

D1. Consider a physical building electrical grid design used with a software aided design system. Describe the following:
(D1a) THREE viewpoints that might be needed to provide a full design description.

**(3 marks)**

| **Answer** |
| --- |
| |

(D1b) FOUR representations with examples that could be used for these design descriptions.

**(2 marks)**

| **Answer** |
| --- |
| |

**Continued...**

D2. Consider the frameworks for mobile applications on the Android system. A very important feature is the access to native APIs (Application Programming Interfaces). Full access to native libraries is possible when developing applications in languages such as Java or Kotlin.

As such, ease of development is a highly subjective matter, these issues might not affect all developers to the same extent. Some issues could have been avoided thanks to the software designers' prior experience, and some might have been encountered due to their inexperience.

Based on the above scenario, answer questions Q-D2a to Q-D2b:

(D2a) Explain TWO reasons in favour of standardizing any form of design description.
**(2 marks)**

| **Answer** |
| --- |
| |

(D2b) Explain THREE reasons against standardizing the same form of description.
**(3 marks)**

| **Answer** |
| --- |
| |

**End of Page.**